Towards Web 3.0: A Unified Development Process for Web Applications Combining Semantic Web and Web 2.0 Technologies

Tzanetos Pomonis[1], Sotiris P. Christodoulou, Andreas B. Gizas

High Performance Information Systems Laboratory (HPCLab),
Computer Engineering & Informatics Department, University Of Patras, Greece

[1]pomonis@ceid.upatras.gr

Abstract

In the last decade many development processes for web applications have been proposed, focusing on discrete web application classes. In this work, we focus on the two most prominent classes of web applications nowadays: Semantic Web and Web 2.0. We attempt to clarify two generic but effective development processes for each one, based on their unique characteristics. Furthermore, as the future web applications, namely Web 3.0, are considered to be the combination of the above classes, we describe a suitable application architecture for Web 3.0 applications together with a proposed hybrid development process in which their specific requirements are taken into consideration. Finally, our findings are presented on the effect of such a development process at the team management aspects of a Web 3.0 application development project.

Keywords

Web 3.0; Web 2.0; Semantic Web; Web Engineering; Web Application; Development Process; 3-Tier Architecture

Introduction

Nowadays, a typical Web application has become a complex and sizable software product which performs advanced business functions. As a result, its development needs to follow a systematic and practical methodology.

This need is even more critical in the case of next generation Web applications, namely Web 3.0, which will combine Semantic Web and Web 2.0 technologies. Although there have been some approaches for such applications (Ankolekar et al., 2007; Leblanc & Abel, 2009), there hasn't been an effort made for a more detailed description of the underlying development process.

In this paper, we propose a practical development process for Web 3.0 applications and in addition, we provide two mature development processes for either Web 2.0 or Semantic Web applications.

Background – Web Application Development

Many of the first Web application Development Processes (shortly WDP) were slight modifications of traditional software ones, as the Web applications initially used to be considered nothing more than a typical software product. In late 90s, Web Engineering was proposed and started to be considered as a discrete domain (Murugesan et al., 1999). Since then the Web application developers took a more specific consideration on the Web distinctiveness, resulting in most of today's WDPs to be heavy extensions of standard software engineering processes.

In Web Engineering, developers manage to make a balance among programming, publishing and business aspects of developing Web applications (Deshpande & Hansen, 2001), and it is obvious that the diversity of web applications is very significant. Thus, researchers have proposed many different WDPs, in order to address discrete characteristics of various web applications.

Initially, the obvious approach was a modified waterfall model (Powell et al., 1998), where the first two stages (planning and requirements analysis) iterated a few times (forming “whirlpools”) in order to clarify several characteristics of the target web application that usually are fuzzy. The iterated waterfall model was soon considered to be too rigid to develop Web applications, in most cases (Pressman, 2005).

The next best-suited process seemed to be the spiral (Boehm & Hansen, 2001). However, there were significant difficulties for its adaptation in Web

application domain, namely the ambiguity in defining the exact steps in each cycle and a common metric to determine the completion of a cycle.

An object-oriented approach for Web application development based on UML is the main concept that powers many processes from Model Driven Design (Mellor & Balcer, 2002) to the Rational Unified Process (Schach, 2005), which is at the same time iterative and object-oriented. The problem is that UML-based design techniques are difficult to be used in practice, because they are often too complicated and time-demanding, since they require a large number of design diagrams and documents, while generally Web applications need to be developed quickly.

The need for quick development pointed to the adoption of Extreme Programming (XP) which seems to be very suitable for Web applications, because of the emphasis on minimum design and quick prototype development (Beck & Fowler, 2001). Even the more general Agile programming techniques are based on reducing the design overhead in project development and having the capability to modify the project development plan (Aoyama, 1998). Nowadays, most web projects adopt such techniques.

As a result of the above, even though there are many different WDP approaches, none of them seems to be a really solid and straightforward solution that can be applied to any Web project. In fact, the only conclusion can be that there are some common features for all Web Engineering processes (Subramanian & Whitson, 2008):

- A typical Web application has much informational content and its development involves considerable attention to publishing issues;
- A WDP should have an information management design, not just a database management design;
- While the application architecture is important, as in Software Engineering processes, the navigational model is often closely related to the application's architecture;
- The Web application process tends to be spiral, agile (if not extreme) and of short duration.

Web 2.0 and Semantic Web Application Development

The abovementioned WDPs try to provide a universal development process for Web applications and don't focus on special requirements of discrete web applications' classes. In this context, we discuss below some more specific approaches, regarding the Semantic Web and Web 2.0 applications which have been the two greatest fields of interest in the Web of the last decade.

Web 2.0 Development Process

The Web 2.0 term (O'Reilly, 2005) represents the adoption of certain technologies and approaches in Web development, targeting more flexible and user friendly applications, and easier distributed collaboration.

Web 2.0 application development principles (Musser & O'Reilly, 2007) call for a lightweight and rapid development process. The complete Web application has to be delivered at end-users in the form of Software as a Service (SaaS), an application software that runs on a Web server rather than being installed on the client computer.

Moreover, Web 2.0 applications are considered to always be in "*Perpetual Beta*" stage (O'Reilly, 2005). Developers are deterred from packaging up new features into monolithic releases, instead they add them on a regular basis as part of the normal user experience, while end-users are engaged as real-time testers.

Based on the above principles of Web 2.0 we have concluded to the minimalistic and abstract development process shown in *Figure 1a*, and by means of it to support the development of Web 2.0 applications.

This process puts the emphasis on keeping the initial design steps to a minimum, on the quick completion of the first "release" of the Web 2.0 application and on facilitating the perceptual update of the application. In this context, the first two steps of our development process, "Requirements Analysis" and "Design" respectively, are kept to a minimum time frame, in order to provide for the desired characteristics of the Web 2.0 application and quickly proceed to the implementation step. Afterward, there is an iterative approach for the next steps, "Implementation", "Testing" and "Maintenance", as there is no solid final product, but simply a "projection" of the Software as a Service to the end-user. At the same time, the usage of up-and-running Web application provides useful

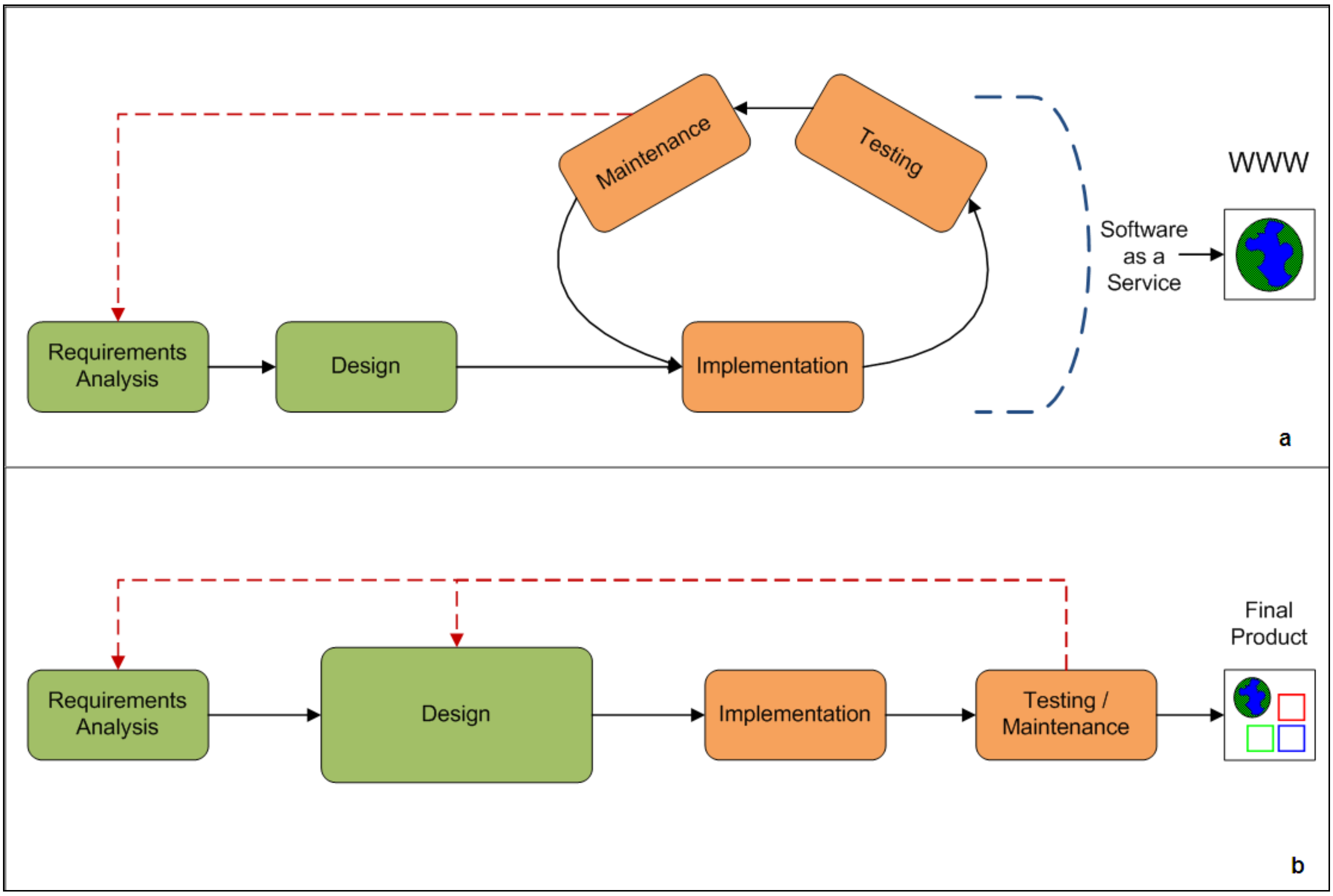

FIG. 1: a) WEB 2.0 DEVELOPMENT PROCESS; b) ONTOLOGY DEVELOPMENT PROCESS

feedback to improve its functionality and usability. Finally, when there is need to extend the current application infrastructure to support some new none-anticipated features which are usually difficult to be added in a really quick and agile way, we consider that the best approach is to re-initiate the whole project, thus starting again from the initial "Requirements Analysis" step and building on the previous design and application code.

Semantic Web Development Process

Semantic Web (Berners-Lee et al., 2001) becomes an innovative technological approach for organizing and exchanging information across applications. The main portions of a conformant application are *Ontologies* and *Reasoning*. An *Ontology* formally defines the concepts (terms and properties) and relationships among them used to describe and represent an area of concern. *Reasoning* appears at the core of the Semantic Web architecture and it is the key component for the derivation of facts expressed inexplicitly in an ontology.

The main idea of designing, developing and supporting a Knowledge Base also affects the whole development process of a Semantic Web application. The development of such application is mainly the development of its core ontological part. Although there are many formal methodologies to develop ontologies (Corcho et al., 2003) (TOVE, KACTUS, METHONTOLOGY, On-To-Knowledge, etc), many prefer a more simplified and intuitive approach for this purpose (Noy & McGuinness, 2001). In short, it is an iterative approach, starting with a rough first pass at the ontology, followed by revision and refining of the evolving ontology and filling in the details.

Our approach to an integrated ontology development process is shown in *Figure 1b*.

We handle the ontology as a stand-alone software product. A moderate "Requirements Analysis" step is followed by a significant "Design" one. Because of its specific nature and complexity, an ontology requires a large "Design" step, making it the most time-demanding step of the process, while the "Implementation" step is short. At "Testing/Maintenance" step, minor deficiencies of the ontology can lead the process back to the "Design"

step, while major flaws shall be confronted by re-initiating the whole process.

Web 3.0 Application Development

Semantic Web and Web 2.0 are not two competitive visions for Web applications, but rather complementary and can be combined together to support the development of next generation Web applications, described by the term Web 3.0 (Lassila & Hendler, 2007; Hendler, 2008). Indeed, these two visions can learn from each other in order to overcome their drawbacks, in a way that enables forthcoming Web applications to combine Web 2.0 principles, especially those that focus on usability, community and collaboration, with the powerful Semantic Web infrastructure, which facilitates the information sharing among Web applications.

However, as described above, Semantic Web and Web 2.0 applications require very different development approaches and infrastructures. In this context, we attempt to analyze the necessary architecture for Web 3.0 applications and propose a suitable development process.

Web 3.0 Supporting Architecture

In *Figure 2* we outline a modular architecture of Web 3.0 applications, introduced in Pomonis et al. (2009), that follows the 3-tier paradigm and extends it in order to comply with the requirements of Web 3.0 applications. This architecture provides the desired physical and logical independence between the discrete Web 2.0 (interface) and semantic (knowledge infrastructure) components which are handled by separate tiers. Especially for the knowledge management components, such an independence provides the flexibility to develop through time by adopting distributed and/or scalable solutions. Moreover, the middle tier is responsible for the interconnection functions and the handling of advanced logical operations.

The lower tier is a knowledge management tier (or system). It integrates and administers data sources, aligning information to a common, mediating ontology; at the same time, this tier performs the low-level reasoning functions that are required in order to deduce implied information.

User requests, queries, additions and other interventions to the ontological model are being interpreted through the application logic tier. It is responsible for the ontological information loading, proper rendering/presentation and the decomposition of the user requests to low-level functions of the knowledge management system. Ontological data and reasoning results are fetched by interaction with knowledge management system, even remotely over the TCP/IP protocol. This interaction is accommodated by a customized distributed version of the OWL-API (Koutsomitropoulos et al., 2010), which overcomes the disadvantages of both DIG 1.1 protocol (lack of full OWL 2 support) and the original OWL-API (only direct in-memory implementation).

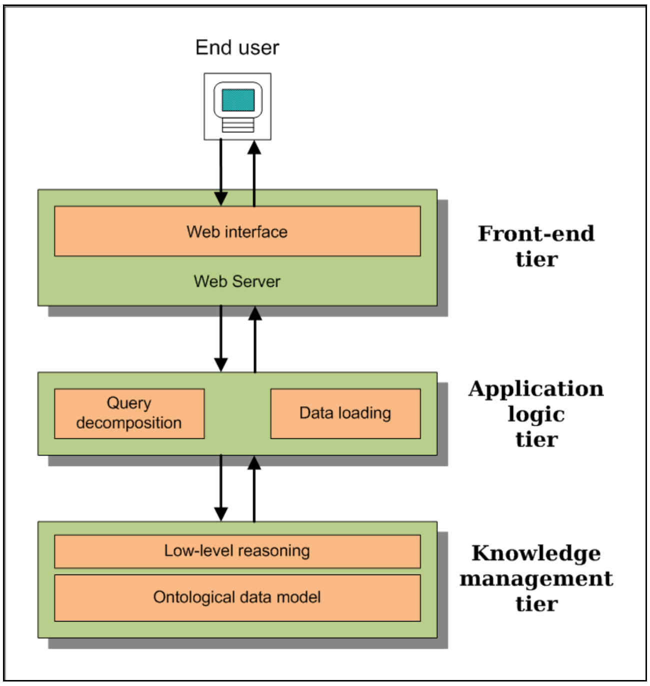

FIG. 2: 3 TIER ARCHITECTURE FOR WEB 3.0 APPLICATIONS

The front-end tier is mainly comprised by a Web server which projects the underlying information to end-users through Web pages, either static ((X)HTML), or dynamic ones, i.e. AJAX oriented PHP pages. Communication with the application tier can be conducted over the HTTP protocol using forms, through XML-based Web services, or even by using specific XML-based network protocols like the PHP/Java Bridge implementation.

Web 3.0 Development Process

Since there is no experience yet in developing Web 3.0 applications, we decided to exploit the great level of independence among the tiers of the abovementioned architecture, and mostly deal with each component/tier as a discrete application regarding its development process (*Figure 3*). The size of each stage is an approximate indication of its demands on time and resources.

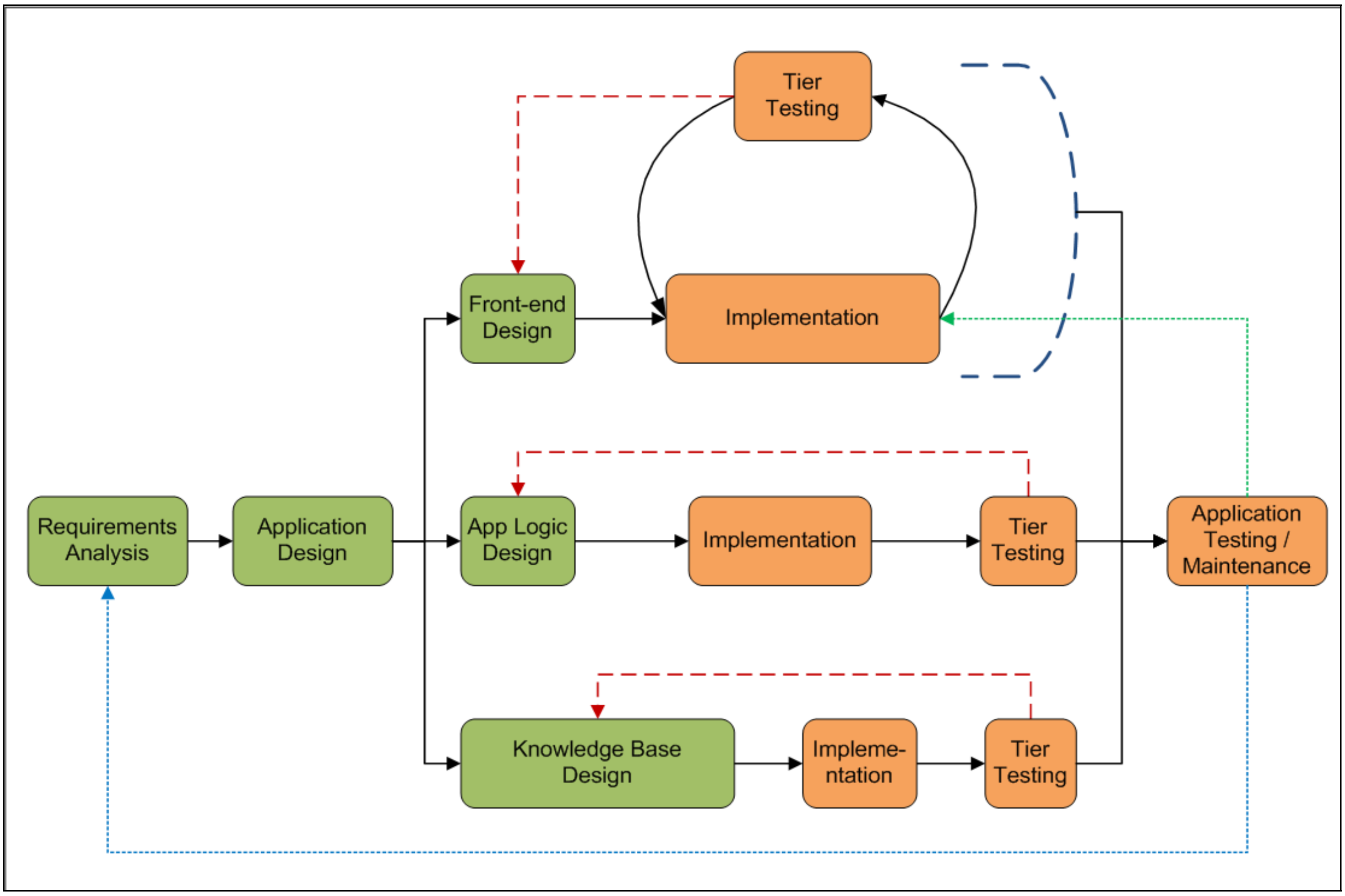

FIG. 3 WEB 3.0 DEVELOPMENT PROCESS

The first step of this approach is a common "Requirements Analysis" where the goals and characteristics of the final application are specified as explicitly as possible.

In "Application Design" stage, the large-scale design of the whole application takes place, where the physical distribution of the three tiers is determined as well as the protocols and tools for their interconnection. A practical approach is to use the customized distributed version of OWL-API for communication between the lower and middle tier, and XML-based network protocols like the PHP/Java Bridge for communication between the top and middle tier. Additionally, several Web 2.0 aspects are addressed during this stage, like the adoption of mash-up technologies or the possibility of user interference with not only the stored information, but also the ontology schema itself. Such decisions could affect all application tiers, thus they have to be taken at this early stage.

After this point, the process is divided into three branches, one for each tier. These three application parts have a great degree of distinctness thus it can be developed almost in parallel:

1) Front-end Tier Development:

a) Front-end Design: Based on the outcomes of "Application Design" stage, the developers make decisions for the frond-end characteristics of the application, such as the content synthesis and rendering as web pages, the design of the user interface and the supported user interactions.

b) Implementation: The majority of Web 3.0 applications will require an interactive and responsive interface, probably AJAX-powered. Nowadays, such interfaces are usually programmed in PHP or JSP.

c) The continuous iteration between Implementation and Testing ensures that all the initial features of the application's front-end are implemented incrementally, but also that additional minor requirements emerging can as well be addressed efficiently.

2) Knowledge Management Tier Development:

a) Knowledge Base Design: this is a rather lengthy stage where the best-suited ontology schema for the content and application is designed,

in a process that can itself be highly iterative, as the produced ontology will act as the fundamental particle of the whole knowledge base.

b) Implementation: This stage involves ontology built and population. In case, the ontology was built during the previous stage using tools like the Protege Ontology Editor, the only action needed is to populate the ontology with data and feed it to a specific reasoning solution (i.e. Pellet or FaCT++).

c) Tier Testing: In this stage, it is important to ensure that the knowledge base can interoperate with the interconnection tool selected in "Application Design" stage.

3)In the middle tier, the sequential "App Logic Design", "Implementation" and "Tier Testing" stages deal with the design and development of the specific programming portions (usually in Java language) that should implement the application's logic features, keeping in mind the inter-tier communication requirements.

Finally, the complete application is tested in "Application Testing/Maintenance" stage with several test use cases, fine tuned where needed, in order to provide a more complete and reliable final product for the end-users. In case of a critical application flaw during one of the tier testing stages, the process falls back to the corresponding tier Design stage. Naturally, in most Web 2.0 and Web 3.0 applications, the front-end features are constantly evolving to address emerged user requirements. If these features do not affect the other tiers, they are usually implemented rapidly in an implementation-testing iteration (see arrow pointing to front-end implementation). Finally, in case of a flaw caused by a bad application design or in case of raised requirements that affect the design of the whole application, the whole process should be re-initiated.

Web 3.0 Project Team Management

One of the great advantages of the above development methodology, mainly due to the parallel tier component processing, comes in matter of team management in a relevant Web 3.0 project.

It has been observed that tier components had different needs in time and people involving in each of their steps. While the front-end component process has a minimal "Tier Design" step and a demanding "Implementation" and "Tier Testing" cycle, the knowledge base system one consists of a rather lengthy and resource-consuming "Tier Design" step and two trivial and lightweight "Implementation" and "Tier Testing" ones. In addition, the application logic component has lightweight "Tier Design" and "Tier Testing" steps, since they have a high degree of correlation with the respective application ones, and a more demanding, but still small regarding the whole application needs, "Implementation" one.

All of the above are summarized in *Figure 4* where all quantities are meant to be relative to the whole project ones.

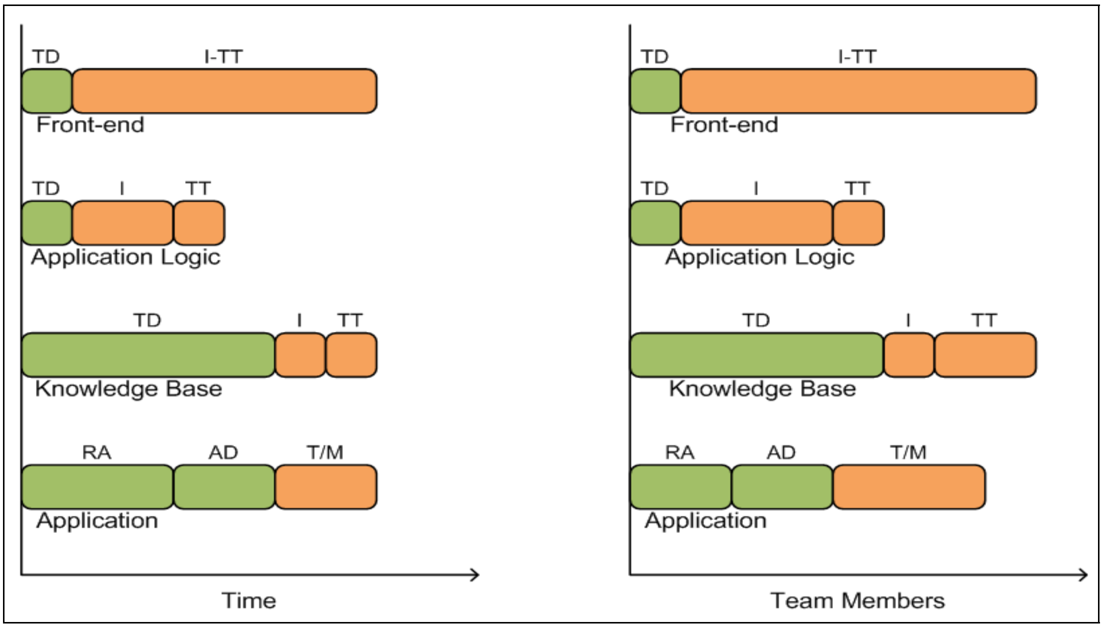

FIG. 4 WEB 3.0 PROJECT NEEDS

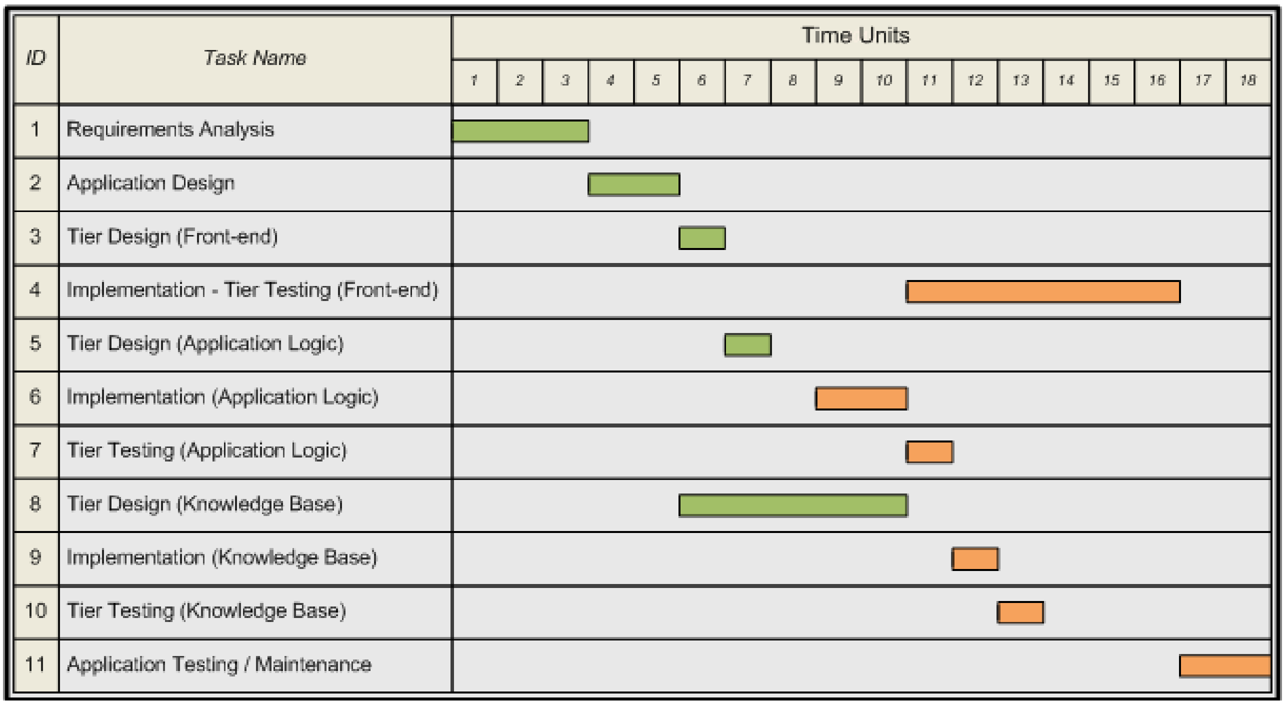

FIG. 5 PROJECT'S CHART

As a result, with proper time scheduling and team member allocation (*Figure 5*), this process can result in decreasing the total project's length, man-months and eventually cost. By its application to a number of small-to-medium scale projects (up to 25 team members) we observed a total time reduction of about 10% and a project team shorten by 25%, compared with the initial prediction based on more traditional WDPs, i.e. spiral model, agile techniques etc.

Indicative Applications

All of the above development processes have been actually applied in several testing and real-world applications in order to derive the conclusions mention in this paper.

Indeed, the discrete Web 2.0 and Semantic Web Development processes were applied both in several R&D Web projects.

Such a Web 2.0 project is "Diazoma" project (http://www.diazoma.gr), where a digital repository was asked to be designed, deployed and supported to store catalogued information for the ancient theaters and odeons in the Greek territory. We made possible for a large group of archaeologists, architects and other ancient theater experts, to add information through a web application about their area of expertise, and afterward we made it available to every visitor of the "Diazoma" website through a friendly and interactive user interface, by developing a Google maps (http://maps.google.com) based navigational application.

Regarding Semantic Web application development, we had participated in design and development of the official University of Patras Institutional Repository (http://repository.upatras.gr/dspace), which shows how to augment traditional digital repository services by the implementation of an extensible semantic search and navigation facility on top of Dspace (http://www.dspace.org). In particular, it is an example how, starting with a semi-structured knowledge model (like the one offered by DSpace), we can end up with inference-based knowledge discovery, retrieval, and navigation among the repository contents.

As for the unified Web 3.0 development process, it was applied in several indicative applications that combine both Semantic Web and Web 2.0 technologies.

Such an indicative application is a semantic movie portal, where information for movies is collected from Internet Movie Database (http://www.imdb.com) using ordinary Web scraping techniques, while information about respective DVD releases is collected from the Amazon website (http://www.amazon.com) using its API. The combined information is then used to populate a single ontology, based on our site's purpose. For each user's search, the relevant results generated by querying the underlying reasoner, are presented through a user friendly interface, providing

complete information for each result and single-point access to it.

Another such application is a semantic book portal (Kalou et al., 2010), where a user can search for any kind of book coming from Amazon, through its API, and also get informed about relevant eBay entries, fetched through the Half eBay API (http://www.half.ebay.com). For each user-triggered search, the desired functionality is provided by real-time searching, firstly through the Amazon API and then, for each intermediate result, through the Half eBay API. The retrieved information is used to create specific instances against a unifying ontology schema that we have constructed in OWL 2. This ontology includes also a series of custom SWRL rules which model a predefined set of user preferences. These rules are fired against the ontological instances, resulting in the further narrowing and classification of results. The final outcome is served through an AJAXised interface which provides complete information for each book, enriched with relevant eBay offers, that is highly customized and adjusted to each user.

Both applications unify retrieved information into an ontology, each time suitable for the particular content characteristics. In this manner, semantically enabled mash-ups have been established, for the movie and the book domain respectively. Thus, an ordinary user of these portals, not only has all the information he needs in a single site, but additionally he benefits from advanced features of semantic personalization (Ankolekar & Vrandecic, 2006; Tziviskou & Brambilla, 2007) and intelligent querying support.

Conclusions

In this work we introduced three WDPs that were indeed applied in several R&D Web projects. The Web 2.0 and Semantic Web ones were used in the development of many real-world projects, while the Web 3.0 WDP was applied in several indicative applications that combine both Semantic Web and Web 2.0 technologies. Each one proved to be suitable for the goals of the corresponding application class, while they were straightforward and simple enough in terms of our needs.

Finally, the proposed Web 3.0 WDP consists of a first attempt to provide a unified development process for Web 3.0 applications, i.e. applications that combine Semantic Web and Web 2.0 Technologies. It has been tested on several web projects and improves the time scheduling and team management aspects of the project (compared to a typical web project), mainly due to the parallel processing of the tier components and their independence, resulting in decreasing the total project's duration, effort and eventually cost.

REFERENCES

Ankolekar, A., Krötzsch, M., Tran, T., & Vrandecic, D. (2007). The two cultures: mashing up Web 2.0 and the Semantic Web. In WWW'07: Proceedings of the 16th International Conference on World Wide Web (pp. 825-834). New York, USA. ACM Press.

Ankolekar, A., & Vrandecic, D. (2006). Personalizing Web surfing with semantically enriched personal profiles. In M. Bouzid and N. Henze (ed.), Proc. Semantic Web Personalization Workshop. Budva, Montenegro.

Aoyama, M. (1998).Web-based agile software development, IEEE Software, November/December 1998, 57-65.

Beck, K., & Fowler, M. (2001). Planning Extreme Programming, Addison-Wesley.

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The Semantic Web. Scientific American, 5.

Boehm, B., & Hansen, W. J. (2001). Understanding the Spiral Model as a Tool for Evolutionary Acquisition. CrossTalk, May 2001.

Corcho, O., Fernandez-Lopez, M., & Gomez-Perez, A. (2003). Methodologies, Tools and Languages for Building Ontologies. Where Is Their Meeting Point?. Data and Knowledge Engineering 46(1): 41–64.

Deshpande, Y., & Hansen, S. (2001) Web engineering: Creating a discipline among disciplines. IEEE Multimedia, April-June, 82-87.

Hendler, J. (2008). Web 3.0: Chicken Farms on the Semantic Web. Computer 41(1), 106-108.

Kalou, A. K., Pomonis, T., Koutsomitropoulos, D., & Papatheodorou, T. (2010). Intelligent Book Mashup: Using Semantic Web Ontologies and Rules for User Personalisation. IEEE Int. Workshop on Semantic Web and Reasoning for Cultural Heritage and Digital Libraries (SWARCH-DL), International Conference on Semantic Computing (ICSC).

Koutsomitropoulos, D. A., Solomou, G. D., Pomonis, T., Aggelopoulos, P., & Papatheodorou, T. S. (2010). Developing Distributed Reasoning-Based Applications for the Semantic Web. In Proc. of the 24th IEEE Int.

Conference on Advanced Information and Networking (AINA 2010)-Int. Symposium on Mining and Web (MAW10). WAINA, pp. 593-598.

Lassila, O., & Hendler, J. (2007). Embracing "Web 3.0". IEEE Internet Computing 11(3), 90-93.

Leblanc, A., & Abel, M. (2009). Linking Semantic Web and Web 2.0 for Learning Resources Management. In Proc. of the 2nd World Summit on the Knowledge Society: Visioning and Engineering the Knowledge Society. A Web Science Perspective (pp. 60 – 69).

Mellor, S., & Balcer, M. (2002). Executable UML: A foundation for model driven architecture. Addison-Wesley.

Murugesan, S., Deshpande, Y., Hansen, S., & Ginige, A. (1999). Web Engineering: A New Discipline for Development of Web-based Systems. In First ICSE Workshop on Web Engineering, International Conference on Software Engineering.

Musser, J., & O'Reilly, T. (2007). Web 2.0 Principles and Best Practices. O'Reilly Media.

Noy, N. F., & McGuinness, D. L. (2001). Ontology development 101: A guide to creating your first ontology. Technical Report SMI-2001-0880, Stanford Medical Informatics.

O'Reilly, T. (2005). What is Web 2.0 – Design patterns and business models for the next generation of software. Retrieved 30 September 2005, from http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html.

Pomonis, T., Koutsomitropoulos, D. A., Christodoulou, S. P., & Papatheodorou, T. S. (2009). Towards Web 3.0: A unifying architecture for next generation web applications. In San Murugesan (ed.), Handbook of Research on Web 2.0, 3.0 and X.0: Technologies, Business and Social Applications (pp. 192-204). IGI Global.

Powell, T., Jones, D. L., & Cutts, D. C. (1998). Web site engineering: Beyond Web page design. Upper Saddle River, NJ: Prentice Hall.

Pressman, R. S. (2005). Software engineering: A practitioner's perspective. New York: McGraw-Hill.

Schach, S., (2005). Object-oriented and classical software engineering. McGraw Hill.

Subramanian, N., & Whitson, G. (2008). Augmented WebHelix: A Practical Process for Web Engineering. In Daniel M. Brandon (ed.), Software Engineering for Modern Web Applications: Methodologies and Technologies (pp. 25-52). IGI Global.

Tziviskou, C., & Brambilla, M. (2007). Semantic personalization of web portal contents. In WWW '07: Proceedings of the 16th International Conference on World Wide Web (pp. 1245-1246). New York, USA. ACM Press.

Tzanetos Pomonis received his B.Sc. from Computer Engineering and Informatics Department of University of Patras in 2003, and M.Sc. in Computational Mathematics & Informatics in Education from Mathematics Department of University of Patras in 2007. He also received a Ph.D. from the University of Patras in 2011. His research interests include Web Engineering, Web 3.0, Web Information Systems, Knowledge Management in the Web, Web 2.0, Artificial Intelligence in the Web, and the Semantic Web.

Sotiris P. Christodoulou received a BSc in computer engineering and informatics from the University of Patras, Greece in 1994. He also received a PhD from the University of Patras in 2004. He is currently a senior researcher of Web engineering at HPCLab at the University of Patras. His research interests include Web Engineering, Hypermedia, Web Information Systems Development, XML. He has participated in several Greek and European R&D projects in the fields of hypermedia technologies since 1994, including: unified cultural information system, numerous WWW infrastructures, intranets, multimedia CDROMs, ESPRIT projects, IST projects, etc. Most projects combined applied research and development, emphasizing on applying cutting-edge technologies to real-world problems.

Andreas B. Gizas is a researcher at the High Performance Information Systems Laboratory (HPCLab), University of Patras. He has received a M.Sc. and a Computer and Informatics Engineer diploma, from the Computer Engineering and Informatics Department. He is currently a PhD candidate of the University of Patras. His research interests include Web Engineering, Hypermedia, Web Information Systems Development, XML and Web 2.0.